A generic approach to variant design in electrical engineering CAD

Dirk Schaefer, Georgia Institute of Technology, USA

This paper proposes a new generic approach to variant design technology in the context of Electrical Engineering Computer-Aided Design (ECAD). Furthermore, a procedure to allow for the implementation of this approach within arbitrary ECAD systems is presented. The technology behind this involves three major steps. Firstly, a variant of an installation is configured on the basis of existing standardized design components. Secondly, a set of commands is automatically compiled that describe the generation of an ECAD project containing the configured components. This is a key novelty, as all these commands are expressed in a non-system specific meta-language, which can then be automatically translated into a macro programming language of any specific ECAD system. Thirdly, the targeted ECAD system can import and process these commands in order to create a corresponding project file in its native data format for further processing.

ECAD/CAE / Variant design / Product configuration / Design automation

1. Introduction

alterations of specific components can be made at any time

area of Computer-Aided Design for Electrical Engineering (ECAD), approaches to develop, generate, and manage design variants of a product in an efficient manner are considered to be key drivers for enabling an increase in productivity going along with a significant cost and time reduction [1]. However, new variants of existing products are more and more often composed of existing basic components (product variant configuration) rather than newly designed. A precondition to allow for this is to have modularized product structures. The degree of modulatization (i.e. the number and complexity of standardized components representing the product structure) finally determine the degree of flexibility and possible variation. In order to handle small variations of standardized components there are two options. An existing standardized component may be altered and added to the database as a specific variant. Alternatively, one-off

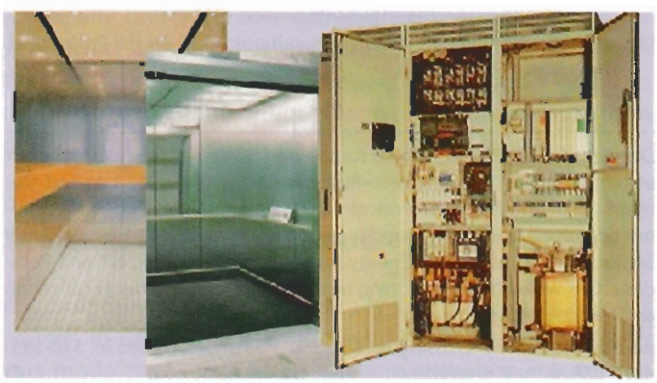

Fig. 1: A typical example of an electrical product variant: elevator and corresponding control cabinet.

tice, is plant engineering and construction. In this sector, the majority of companies developing electrical installations tend to re-use their existing designs, plans and project documentations to develop additional customer specific variants [2]. A typical example of such a variant would be a product family of similar elevators and their corresponding control cabinets (see Fig. 1).

Unfortunately, this process of re-using existing designs for the generation of further variants is still predominantly performed in a manual fashion. Obviously, this is both far from being efficient and effective, and is error-prone [1]. Consequently, due to the permanently increasing pressure of competition in the market it is vital to develop automatisms that facilitate the computer-aided generation of product variants.

As alluded to earlier, the purpose of this paper is to propose a new generic approach to variant design technology that may be deployed in future ECAD systems. The fundamental idea behind the approach is to automate the natural workflow process of creating ECAD design variants as it is manually performed by most companies today. As such, the approach presented is closely related to industrial best practice and derived from day-to-day operations.

Over the past twenty years, many people have worked in the area of variant design. However, this topic has predominantly been addressed with regard to the geometry-driven mechanical engineering context. A detailed overview of key works in the field is given in [2]. As for variant design in the electrical engineering CAD context, very little has been published. An outline of the scope and potential benefits of new approaches to variant design in ECAD is given in [1].

2. The basic concept as applied in practice

Many companies approach their potential customers by technical field sales and distribution staff. These sales persons usually have selling catalogues on their disposal, which allow customers to assemble bespoke product variants based on basic components that can be combined. Hereby, the number of components that can be chosen from (at this stage of the sales process) tends to be relatively small. Once such a rough pre-configuration is finished, design engineers usually determine a resultant technical fine configuration. This fine configuration comprises all parts and components required to make up the complete product variant desired and may consist of thousands of items [2, 3].

The next step in creating the configured product variant requires a design engineer to use an ECAD system, generate a new (empty) project file, copy the components configured into the project and specify individual customer and order details. Subsequently, alterations necessary to the specific project may be accomplished and the process of generating an updated ECAD project documentation according to these alterations or amendments made has to be initiated.

In order to make the development of product variants more effective, ECAD system vendors aim towards an automatic computer-aided support of the workflow process outlined above [4].

3. Functional principle

The basic functional principle of the generic variant design approach proposed in this paper is now described. It is based on aspects from knowledge-based product configuration [3, 5], the programming of design variants, as well as parametric product modelling and process automation [1, 2]. The fundamental idea behind the approach is to automatically generate an entire technical documentation of an electrical installation on the basis of a placed order specification. The overall process to achieve this involves five steps as shown in Fig. 2.

- Firstly, a design variant of an installation is configured by composing standardized modules of existing (previously developed) components.
- Secondly, all components identified to compose a specific variant are stored in a data file based on a bespoke data structure that describes ECAD design projects in general.
- Thirdly, a set of commands describing the generation of a typical ECAD project containing all the components configured together is automatically compiled.
- Fourthly, the above variant project description expressed in non system specific meta-commands is automatically translated into commands of a specific macro programming language associated to a particular ECAD system.
- Fifthly, the specific ECAD system targeted can import and process these commands to create a corresponding project that may be handled or dealt with in any way the ECAD system allows.

4. Technology approach

An overview of a technical approach to realize the functional principle explained in section 3 is described and illustrated in Fig. 3.

Step one of the functional principle (see Fig. 2) requires a software tool to manage, browse, identify and select already

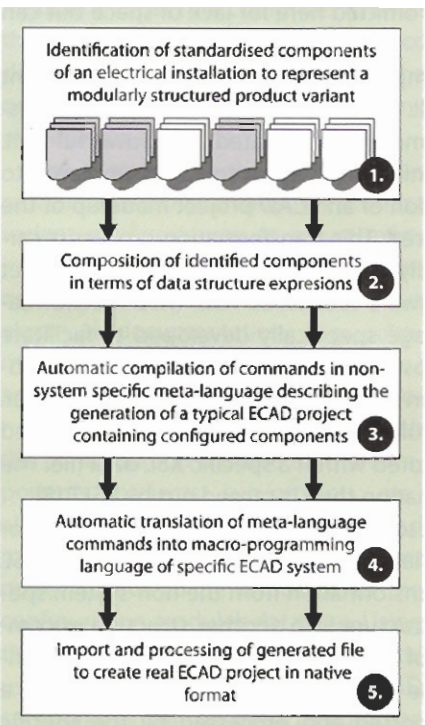

Fig. 2: Functional principle of generic ECAD variant design approach.

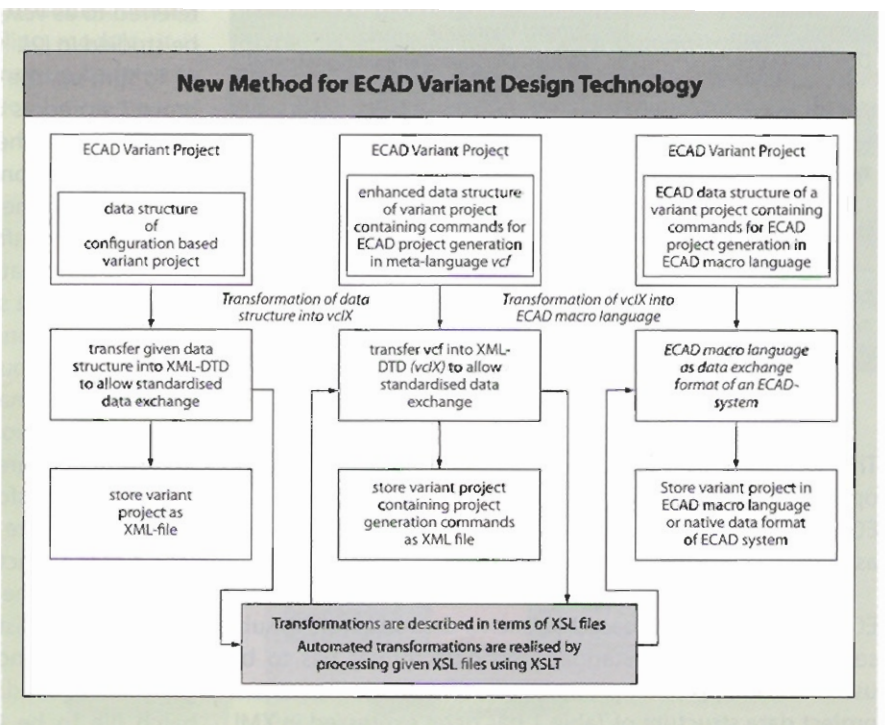

Fig. 3: Realization of approach to automated generation of ECAD/ECAE project documentations.

existing design components. Product data management systems (PDM) may be used for this purpose. Alternatively, a variety of commercial product configuration tools are available on the market, and even low cost table-based solutions based on e.g. Microsoft-Excel™/Access™ may be deployed if appropriate in terms of complexity [6].

As already mentioned, a data structure specifically tailored for representing configuration based variant projects is required. Such a data structure primarily has to cover components and documents typically used to make up an entire ECAD project documentation. An extract of a simplified data structure developed for this purpose and used for the implementation of a software prototype of the approach proposed in this paper is shown in Table 1. It is expressed in Extended Backus Naur Form (EBNF) [7], using the following terminology:

- '=' means is defined as.
- '.' means end of definition.
- '[ ]' means optional, i.e. never or once.
- '{ }' means optional, but one or more times.
- '( )' is just bracketing.

Table 1: Simplified data structure for configuration-based ECAD variant projects expressed in EBNF.

Project_Lib	=	Project {Project}.
Project	=	Project_Attributes Part_Libs Symbol_Libs Doc_Templates Documents .
Project_Attributes	=	Project_Name [Project_Reference] [Drawing_Number] [Order_Number] [Project_Description] [Customer] [Customer_Address] [Contact_Person] [Telephone] .
Part_Libs	=	Part_Lib_Vendor {Part_Lib_Vendor}.
Part_Lib_Vendor	=	Part {Part}.
Symbol_Libs	=	Symbol_Lib { Symbol_Lib }.
Symbol_Lib	=	Symbol {Symbol}.
Doc_Templates	=	Ladder_Diagram_Templates Terminal_List_Templates .
Ladder_Diagram_Templates	=	Ladder_Diagram_Template {Ladder_Diagram_Template}.
Terminal_List_Templates	=	Terminal_List_Template {Terminal_List_Template}.
Documents	=	Ladder_Diagrams Terminal_Lists .
Ladder_Diagrams	=	Ladder_Diagram {Ladder_Diagram}.
Ladder_Diagram	=	Symbol {Symbol} {Connection}.
Terminal_Lists	=	Terminal_List {Terminal_List}.
Connection	=	Coordinate Coordinate [Coordinate].

The simplified data structure shown in Table 1 was developed in consultation with a German vendor of a commercial ECAE system. For ease of presentation trivial data types, such as *Integer*, *Real*, *Char*, *String*, etc. have been omitted.

In order to further process and automatically evaluate ECAD variant projects based on such a data structure in subsequent processes, a standardized data format has to be used. In the approach presented in this paper the variant project data structure of Table 1 has been expressed in XML (eXtensible Mark-up Language). Simply speaking, XML is a language for describing hierarchically composed objects that distinguishes between the structure of objects and their content and layout [8].

The XML counterpart of the above data structure is a so-called XML Document Type Definition (DTD). A DTD is a collection of XML markup declarations that define the structure, elements and attributes that can be used in a document that complies with the DTD. By consulting the DTD a parser can work with the tags from the markup language that document uses [10]. For a graphical representation of the DTD corresponding to the data structure of Table 1 as well as the corresponding DTD source code the interested reader may refer to [2].

Realizing step three of the functional principle (see Fig. 2) involves formulating commands expressing the generation of a configuration based ECAD project in a non-system specific form. Hence, a suitable meta-language has been developed and applied. This meta-language is called *variant construction language* (*vcl*) and covers a variety of system commands similar to those being typical for contemporary ECAD system's macro languages [2]. It has been proven that vcl is a regular language, i.e. of Chomsky-Type-3 [7, 9]. An EBNF representation of a simplified version of the prototypic vcl is presented in Table 2. This is by no means complete and only meant to demonstrate the realization principle.

Following the functional principle presented, a variant project stored as XML file now can be transferred into a new data format describing the generation of an ECAD project containing the configured components. Technically speaking, this means to enhance the original XML data structure of a configuration based variant project in such a way that it allows to model and express ECAD system commands in a non-system specific language. Again, the XML counterpart of the grammar shown in Table 2 (i.e. a corresponding DTD referred to as *vclX*) is omitted here for lack of space but can be studied in [2].

To sum-up, the components of a configured ECAD variant project stored as XML file have been picked-up and transferred into another, more sophisticated and powerful XML data structure containing non system specific commands to describe the generation of an ECAD project made up of the components configured. This transformation process is carried out automatically using XSL (eXtensible Style sheet Language) and a software tool called XSLT (XSL Transformation). XSL is a language specifically developed to facilitate transformation purposes and allows defining rules describing the transformation from one XML structure into another. The transformation rules necessary to perform the desired transformation are stored within a specific XSL data file. The actual data transformation then is carried out by XSLT [8].

The procedure described above analogously recurs for step 4 of the functional principle. However, this time an XSL file describing the transformation from the non-system specific command list structure into another structure encompassing commands of a specific ECAD system's macro language is required. The result of this final transformation is a batch file to be imported and processed by the specific ECAD system chosen. In other words, a real ECAD project in a native data format has been created.

Table 2: Grammar of the variant construction language vcl in EBNF.

S	=	METALANGUAGE
METALANGUAGE	=	(PROJECTNAME (ATTACH | IMPORT | OPEN | CLOSE | CREATE | MODIFY)) { METALANGUAGE } ■
ATTACH	=	"ATTACH" NAME [PROJECT | (PROJECT SUBPROJECT)] ■
IMPORT	=	"IMPORT" NAME [PROJECT | (PROJECT SUBPROJECT)] ■
OPEN	=	"OPEN" (PROJECT | (PROJECT SUBPROJECT))[DOCUMENT | TEMPLATE] ■
CLOSE	=	"CLOSE" (PROJECT | (PROJECT SUBPROJECT)) [DOCUMENT | TEMPLATE] ■
CREATE	=	"CREATE" (PROJECT | (PROJECT SUBPROJECT)) [DOCUMENT | TEMPLATE] ■
MODIFY	=	"MODIFY" (PROJECT | (PROJECT SUBPROJECT)) [DOCUMENT | TEMPLATE] ■
PROJECT	=	"PROJECT" NAME { PARAMETER }■
SUBPROJECT	=	"SUBPROJECT" NAME ■
DOCUMENT	=	"DOCUMENT" NAME [SYMBOL | CONNECTION]{ PARAMETER }■
TEMPLATE	=	"TEMPLATE" NAME [SYMBOL | CONNECTION] { PARAMETER }■
SYMBOL	=	"SYMBOL" SYMBOLNAME LIBNAME COORDINATE {PIN} { PARAMETER }■
CONNECTION	=	("ECONNECT" | "MCONNECT") COORDINATE COORDINATE [COORDINATE]■
PARAMETER	=	NAME VALUE■
PIN	=	PIN_NAME COORDINATE■
COORDINATE	=	X_POS Y_POS■

5. Software module conception

An architecture of a prototype ECAD variant software module for realizing the variant design technology approach proposed in this paper is now presented. The overall variant module architecture comprises of two sub-modules, namely the configuration module and the coupling module. It is developed as a self-contained unit that may be coupled to one or more ECAD systems rather than directly implemented within a specific system (see Fig. 4).

The purpose of the configuration module is to either create new product variant configurations or adjust existing ones using existing basic components. Hereby, the various plans of an electrical documentation (e.g. schematics, terminal plans, part lists, etc.) are drawn on as configuration components. Prior to being able to work with the configuration module all the components (projects, sub-projects, etc.) already existing on ECAD site and meant to be available for variant projects have to be imported to the configuration module's database. Using the features of a comfortable graphical user interface (GUI), the imported ECAD components can be used to combine new variant projects, to adjust previously created design variants and to add further basic components to the database. The design knowledge with regard to constraints describing possible combinations and configurations has to be brought into the database as well. The system kernel of the configuration module therefore has to incorporate an intelligent mechanism to maintain, check and control the compliance of the constraints modelled. Due to the complexity of knowledge based configuration systems (i.e. expert systems from a Computer Science perspective) [10] the development of proprietary knowledge based configuration tools is strictly not recommended. There are many commercial solutions for almost any configuration tasks and complexity available on the market as pointed out earlier [5],[6].

The purpose of the coupling module is to automate the steps of the workflow process that today are usually performed manually by an engineer once the components for a variant configuration have been determined. A first task for the coupling module is to import a variant configuration from the configuration module. Subsequently, it has to create a file of non-system specific commands describing the relevant steps to open a new ECAD project and to include the configured components. After that, the coupling module has to transfer these non-specific commands into a data file to be imported by a specific ECAD system for further processing.

6. Conclusion and further research

Currently both ECAD users and vendors prefer variant design approaches that allow for automated generation of complete project documentations based on configuration using modular standardized components [4]. The novel generic approach to automated product variant design technology presented in this paper has been realized as a software prototype. Its applicability has been demonstrated using an industrial problem, and has been found to bear a high potential in respect to cost and time reduction in the area of electrical computer-aided design and engineering [2]. An

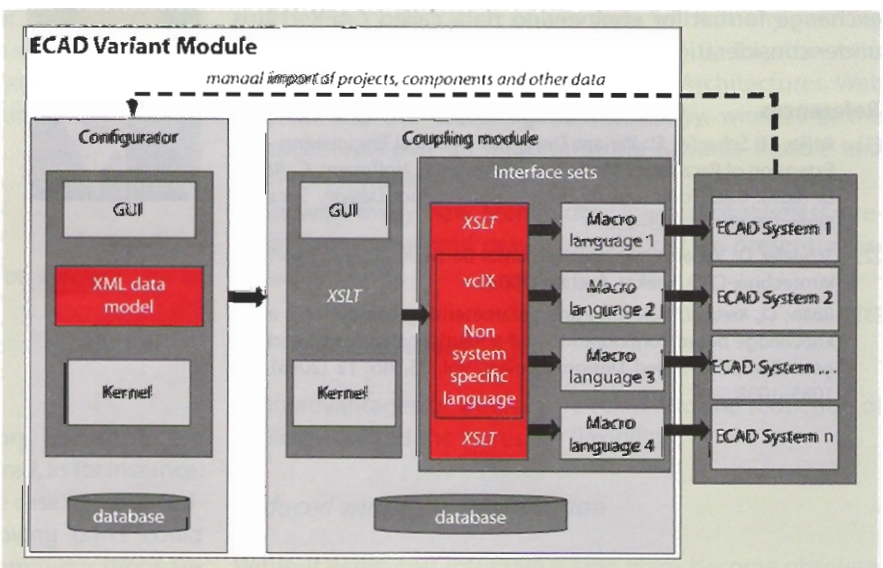

Fig. 4: System architecture of the prototype ECAD variant module.

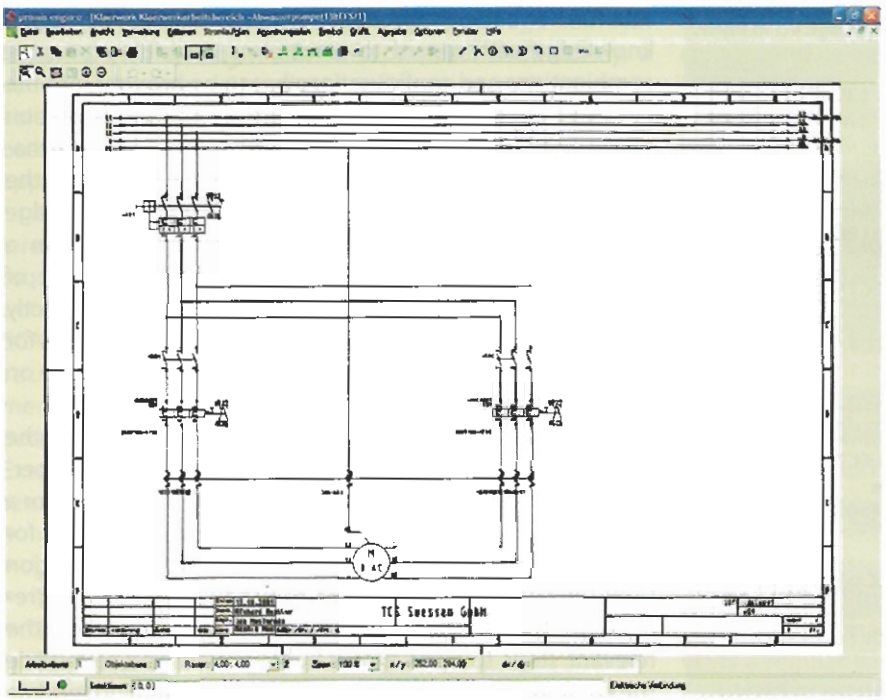

Fig. 5: Example of an ECAD schematic (ladder diagram) automatically generated using the prototype variant module.

example of a schematic automatically generated using the software prototype of the variant module is shown in Fig. 5. The results produced using the variant design approach as described in Fig. 1 and Fig. 2 are basically large, computer-generated code files in XML and ECAD specific macro languages, respectively. The interested reader may refer to [2] for a complete and detailed example.

Current work aims at a system integration realization based on the international product data exchange standard STEP (STandard for the Exchange of Product model data) [11]. Particularly the Application Protocol 'AP 212: Electrotechnical Design and Installation' is of interest here. It was published in 2001, and some major ECAD vendors have recently made corresponding translators available. In addition, recently developed industry-driven neutral data exchange format for engineering data called CAEX [12] is under consideration to be used for the same purpose.

References

[1] *Roller, D., Schaefer, D.*: Variant Design in Electrical Engineering – An Extension of Parametric Modeling. In: Brunet, P., Hoffmann, C., Roller, D. (eds.): CAD Tools and Algorithms for Product Design, Springer, Berlin (2000), pp. 233–250.

[2] *Schaefer, D.*: Variantentechnologie unter Beruecksichtigung von Elektrotechnik-CAD, Shaker, Aachen (2003).

[3] *Roller, D., Kreuz, I.*: Selecting and parameterising components using knowledge based configuration and a heuristic that learns and forgets. Computer-Aided Design, Elsevier, Vol. 35, no. 12 (2003), pp. 1085–1098.

[4] *Baguley, P., Schaefer, D., Page, T.*: Costing Issues Regarding Product Variant Design. In: Maropoulos, P.G., Schaefer, D. (Eds.): ECAD/ECAE 2004 – Proceedings of the 1st International Conference on Electrical/Electromechanical Computer-Aided Design & Engineering, Durham, England (2004), pp. 41–45.

[5] *Guenter, A. and Kuehn, C.*:, Knowledge-based Configuration – Survey and Future Directions. In: Puppe, F. (ed.): Knowledge-based Systems, Springer, Berlin (1999), pp. 47–66.

[6] *Mesina, M., Roller, D.*: Configuration Management using standard tools. In: Yan, X.-T., Jiang, Ch.-Y., Juster, N.P. (eds.): Perspectives from Europe and Asia on Engineering Design and Manufacture, Kluwer Academic Publishers (2004), pp. 645–658.

[7] *Brookshear, J.G.*: Computer Science: An Overview. Addison Wesley (2004).

[8] *Skonnard, A., Gudgin, M.*: XML Quick Reference – A Programmer's Reference to XML, XPath, XSLT, XML Schema, Soap and More. Addison Wesley (2001).

[9] *Roller, D., Dettlaff, B., Richert, U.*: Rationalization in the Electrical Engineering Process. In: Roller, D. (Ed.): Mechatronics – Advanced Development Methods and Systems for Automotive Products, ISATA 1996 Proceedings, Croydon, England (1996), pp. 279–284.

[10] *Sriram, R.D.*: Intelligent Systems for Engineering: A Knowledge-based Approach. Springer (1997).

[11] *Pratt, M.J.*: ISO 10303, the STEP standard for product data exchange, and its PLM capabilities. International Journal of Product Lifecycle Management, vol. 1, no. 1 (2005), pp. 86–34.

[12] *Fedai, M., Drath, R.*: CAEX – A neutral data exchange format for engineering data. atp International – Automation Technology in Practice, vol. 3, no. 1 (2005), pp. 43–51.

Received: April 13, 2006.

Dirk Schaefer (Dr., 36) is currently an Assistant Professor in the Woodruff School of Mechanical Engineering at the Georgia Institute of Technology, USA. Prior to this he was a University Lecturer in the School of Engineering at Durham University, UK. Dirk obtained his Ph.D. in Computer Science from the University of Stuttgart, Germany. He has published around fifty papers on CAD, CAM, CAE and ECAD in conference proceedings, journals and books. Dirk's current research interests are focused on the high-impact interdisciplinary area of Information Engineering of Complex Engineered Systems.

Address: George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, 210 Technology Circle, Savannah, Georgia 31407, USA, e-mail: dirk.schaefer@me.gatech.edu